

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
6 December 2007 (06.12.2007)

PCT

(10) International Publication Number
WO 2007/139552 A1

(51) International Patent Classification:
H04L 29/08 (2006.01)

(21) International Application Number:
PCT/US2006/021067

(22) International Filing Date: 31 May 2006 (31.05.2006)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant (for all designated States except US): **CITRIX SYSTEMS, INC.** [US/US]; 851 West Cypress Creek Road, Fort Lauderdale, FL 33309 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **MIRANI, Rajiv** [IN/US]; 106 Walnut Hill Ct., Los Gatos, CA 95032 (US). **WONG, Stanley, Hang** [US/US]; 7565 Bollinger Road, Cupertino, CA (US). **CHAUHAN, Abhishek** [IN/US]; 107 Rio Vis, Los Gatos, CA 95032 (US).

(74) Agent: **MCKENNA, Christopher, J.**; CHOATE, HALL & STEWART, Two International Place, Boston, MA 02110 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

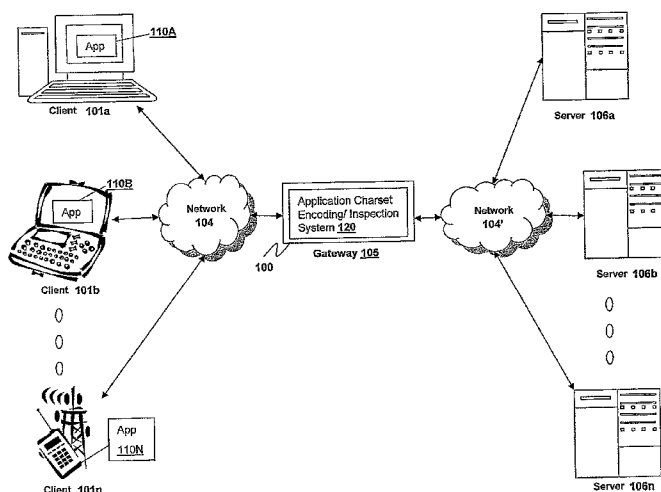
(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— with international search report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEMS AND METHODS FOR DETERMINING THE CHARSET ENCODING FOR DECODING A REQUEST SUBMISSION IN A GATEWAY



(57) Abstract: The systems and methods are disclosed for determining the charset encoding of a request submission intercepts by a gateway (105) to decode the request according to the charset. A gateway (105) receives a request from a client (101a,... / 101n) comprising encoded content, such as a url encoded request. The gateway (105) determines the application (110A,..., 110N) generating or associated with the request. Based on the determined application, the gateway identifies the character encoding scheme associated with or to be used for the application. The gateway (105) then decodes the portion of the request using the identified character encoding scheme and applies any rules or policies to the request. In some embodiments, the gateway operates as an application firewall or security control system that applies policies to encoded application network traffic to which it can decode according to the encoding scheme associated with each application.

SYSTEMS AND METHODS FOR DETERMINING THE CHARSET ENCODING FOR DECODING A REQUEST SUBMISSION IN A GATEWAY

Technical Field

The present invention generally relates to determining the charter encoding for a request of an application. More particularly, the present invention relates to systems and methods for determining the charset encoding for decoding a request of an application submitted via a gateway.

Background Information

A commonly known way of harvesting information from users on the world-wide computer network known as the Internet is a technique known as a “form,” which is a page that provides one or more fields for the user of the browser to populate. However, there exist many different encodings for characters that may be submitted via a form and there may be no way to indicate, in the request, which charset should be used for the request. Use of a character encoding set (“charset”) to decode a form submission that is different from that used to prepare it can result in rejection or misinterpretation of a form by a server. This problem is exacerbated for an intermediate network device, such as an application firewall deployed between the server and the client because the network device has no access to the application logic responsible for generating and processing the forms.

One technique that has attempted to solve this charset problem relies on the network device recording the charset used to generate and send the form to the client. The technique, however, will not work when the form is entirely generated on the client using, for example, Javascript. The network device may also attempt to solve

this problem by assuming that all form requests use an encoding known as “utf-8.”

This approach has obvious drawbacks.

Summary of the Invention

The systems and methods of the present invention provide a solution for efficiently and robustly handling, decoding, and analyzing requests from applications that may comprise differently encoded content or content encoded with an unidentified charset without requiring the server application or the client browser to be recoded. The described techniques may also be extended to allow policies to be applied to the requests in order to use different charsets for different situations. As a result, form submissions and other requests between servers and clients may be inspected to ensure that no malicious requests (such as SQL injection requests) are allowed to reach the server application with a minimum number of false positives.

In one aspect, the present invention is related to method for determining the character encoding used to use to decode a request. The method includes receiving a request and determining the application program of multiple application programs the request corresponds. The method identifies a character encoding associated with the determined application program, and uses the identified character encoding to inspect the request.

In one embodiment, the method determines to which one of the multiple application programs the request corresponds from an attribute of the request. In some embodiments, the attribute of the request includes: 1) a source identifier, 2) a destination identifier, 3) a port identifier, 4) a protocol identifier, 5) header information or a 6) Uniform Resource Locator address. In another embodiment, the

method determines to which one of the multiple application programs the request corresponds using a cookie included in the received request.

In other embodiments, the method identifies the character encoding associated with the determined application program using a file containing associations between character encodings and applications. In some embodiments, the method identifies the character encoding associated with the determined application program using a database containing associations between character encodings and applications.

In one embodiment, the method includes receiving a second request, determining a second one of the multiple application programs to which the second request corresponds, and identifying a second character encoding associated with the determined second application program. In another embodiment, the method includes receiving a request generated by a client. The method may determine to which one of the multiple application programs the request corresponds using an attribute of the client. In yet another embodiment, the method includes receiving a request based on a cached form page.

In another aspect, the present invention is related to a gateway capable of determining the character encoding to be used to decode a request. The system includes a receiver in communication with a client via a network and receiving a request from the client. The system also includes a character set engine in communication with the receiver. The character set engine identifies a character encoding associated with a received request responsive to the application to which the request is directed and uses the identified character set to inspect the request.

In one embodiment, the gateway communicates with multiple clients. In some embodiments, the gateway uses one of the following contained in the request to determine an application program to which the request is directed: 1) a source

identifier; 2) a destination identifier, 3) a port identifier, 4) a protocol identifier, 5) header information, or a 6) Uniform Resource Locator address. In another embodiment, the character set engine includes a database associating character encodings and applications. In other embodiments, the character set engine includes a file associating character encodings and applications.

In yet another aspect, the present invention is related to a method for inspecting by a gateway a client request having an encoded portion. The method includes receiving, by the gateway, a request from an application program on a client. The method also includes determining, by the gateway, to which one of a plurality of application programs the request corresponds, and identifying a character encoding associated with the determined application program. The method further includes decoding, by the gateway, a portion of the request using the identified character encoding, and inspecting or analyzing the decoded portion of the request.

In one embodiment of the method, the gateway determines to which one of the plurality of application programs the request corresponds using an attribute of the request. In another embodiment, the gateway determines to which one of the plurality of application programs the request corresponds using an attribute of the client. In yet another embodiment, the gateway applies a policy to the request based on inspection of the decoded portion of the request.

In some embodiments, the method includes receiving, by the gateway, a second request from a second application program on one of the client or a second client. The gateway determines which one of the plurality of application programs the second request corresponds, and identifies a second character encoding associated with the determined application program. The gateway decodes a portion of the second request using the identified second character encoding. In some embodiments,

the gateway inspects or analyzes the decoded portion of the second request. In another embodiment, the method includes applying, by the gateway, a policy to the second request based on inspection of the decoded portion of the second request.

The details of various embodiments of the invention are set forth in the accompanying drawings and the description below.

Brief Description of the Drawings

The foregoing and other objects, aspects, features, and advantages of the invention will become more apparent and may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

FIG. 1A is a block diagram of an example network environment deploying a gateway having a system to determine the charset encoding for an application;

FIG. 1B is a block diagram of an another network environment of deploying on a client and/or a server a system to determine the charset encoding for an application;

FIGs. 1C and 1D are block diagrams of embodiments of a computing device for practicing an illustrative embodiment of the system of the present invention;

FIG. 2 is a block diagram of a system for determining the charset encoding for an application to use for decoding and analyzing a request from a client; and

FIG. 3 is a flow diagram of steps performed in practicing an embodiment of a technique to determine the charset encoding for an application to use for decoding and analyzing a request of a client.

The features and advantages of the present invention will become more apparent from the detailed description set forth below when taken in conjunction with

the drawings, in which like reference characters identify corresponding elements throughout. In the drawings, like reference numbers generally indicate identical, functionally similar, and/or structurally similar elements.

Description

Certain illustrative embodiments of the present invention are described below.

It is, however, expressly noted that the present invention is not limited to these embodiments, but rather the intention is that additions and modifications to what is expressly described herein also are included within the scope of the invention. Moreover, it is to be understood that the features of the various embodiments described herein are not mutually exclusive and can exist in various combinations and permutations, even if such combinations or permutations are not expressly made herein, without departing from the spirit and scope of the invention.

FIG. 1A depicts a block diagram of a network environment having a gateway deploying an application charset encoding and inspection system 120. As shown in FIG. 1A, the example network environment includes a plurality of clients 101a-101n, a plurality of servers 106a-106n., and a gateway 105, which may also referred to as an appliance, gateway appliance, gateway server or gateway device. The servers 106a-106n manage applications, databases, and other information systems that provide requested content to the clients 101a-101n. Each of the clients 101a-101n and servers 106a-106n may be any type and form of computing device, such as the computing device 100 described in more detail below in conjunction with FIGs. 1C and 1D. For example, any of the client 101a-101n may be a mobile computing device, such as a telecommunication device, e.g., cellphone or personal digital assistant, or a laptop or notebook computer in addition to any type of desktop computer.

Each of the clients 101a-101n are communicatively coupled to gateway 105 via a network 104, while gateway 105 is communicatively coupled to servers 106a-106n via a network 104'. In one embodiment, network 104 comprises the Internet and network 104' comprises a private data communication network such as a corporate or enterprise network. The networks 104, 104' can be any type and form of network, public, private or otherwise, and in some cases, may be the same network.

Although FIG. 1 shows a network 104 and a network 104' between the clients 101a-101n and the servers 106a-106n, the clients 101a-101n and the servers 106a-106n may be on the same network 104 or 104'. The networks 104 and 104' can be the same type of network or different types of networks. The network 104, 104' can be a local-area network (LAN), such as a company Intranet, a metropolitan area network (MAN), or a wide area network (WAN), such as the Internet or the World Wide Web. The network 104, 104' may be any type and/or form of network and may include any of the following: a point to point network, a broadcast network, a wide area network, a local area network, a telecommunications network, a data communication network, a computer network, an ATM (Asynchronous Transfer Mode) network, a SONET (Synchronous Optical Network) network, a SDH (Synchronous Digital Hierarchy) network, a wireless network and a wireline network. The topology of the network 104, 104' may be a bus, star, or ring network topology. The network 104, 104' and network topology may be of any such network or network topology as known to those ordinarily skilled in the art capable of supporting the operations described herein.

As shown in FIG. 1A, the gateway 105 is deployed between a first network 104, such as a public data communication network, and a second network 104' such as a private data communication network. In other embodiments, the gateway 105

may be located on the first network 104 or on the second network 104'. In other embodiments, the gateway 105 could be an integral part of any individual client 101a-101n or any individual server 106a-106n on the same or different network 104 as the client 102a-102n. As such, the gateway 105 may be located at any point in the network or network communications path between a client 101a-101n and a server 106a-106n.

Each of the clients 101a-101n may execute, operate or otherwise provide an application 110a-110n, generally referred to herein as application or application 110. The application 110 can be any type and/or form of software, program, or executable instructions such as any type and/or form of web browser, web-based client, client-server application, a thin-client computing client, an ActiveX control, or a Java applet, or any other type and/or form of executable instructions capable of executing on client 101a-101n. In some embodiments, the application 110a-110n may be a server-based or a remote-based application executed on behalf of the client 101a-101n on a server 106a-106n. In one embodiment the server 106a-106n may display output to the client 101a-101n using any thin-client or remote-display protocol, such as the Independent Computing Architecture (ICA) protocol manufactured by Citrix Systems, Inc. of Ft. Lauderdale, Florida or the Remote Desktop Protocol (RDP) manufactured by the Microsoft Corporation of Redmond, Washington.

In some embodiments, the server 106a-106n or a server farm may be running one or more applications 110, such as an application providing a thin-client computing or remote display presentation application. In one embodiment, the server 106a-101n or server farm executes as an application 110, any portion of the Citrix Access Suite™ by Citrix Systems, Inc., such as the MetaFrame or Citrix Presentation Server™, and/or any of the Microsoft® Windows Terminal Services manufactured by

the Microsoft Corporation. In one embodiment, the application 110 is an ICA client, developed by Citrix Systems, Inc. of Fort Lauderdale, Florida. In other embodiments, the application 110 includes a Remote Desktop (RDP) client, developed by Microsoft Corporation of Redmond, Washington. In other embodiments, the server 106a-106n may stream an application 110 to a client 101a-101n. Also, the server 106a-106n may run an application 230, which for example, may be an application server providing email services such as Microsoft Exchange manufactured by the Microsoft Corporation of Redmond, Washington, a web or Internet server, or a desktop sharing server, or a collaboration server. In some embodiments, any of the applications 110 may comprise any type of hosted service or products, such as GoToMeeting™ provided by Citrix Online Division, Inc. of Santa Barbara, California, WebEx™ provided by WebEx, Inc. of Santa Clara, California, or Microsoft Office Live Meeting provided by Microsoft Corporation of Redmond, Washington.

In accordance with one embodiment, the gateway 105 includes an application charset encoding and inspection system 120. As will be described in further detail below, this system 120 receives a request from a client 101a-101n comprising encoded content. For example, the client 101 may submit an HTTP form or request having encoded content, such as a url encoded portion. In one case, the type of encoding scheme may not be known from the request. The system 120 determines the application generating or associated with the request. For example, the system 120 may identify from the request an internet protocol address and/or port that is associated with an application. Based on the determined application, the system 120 identifies the character encoding scheme associated with or to be used for the application. For example, the system 120 may lookup the encoding scheme for a database, configuration information, or from a policy engine. Then, the system 120

decodes the portion of the request using the identified character encoding scheme and applies any rules or policies to the request. In some embodiments, the system 120 operates as an application firewall or security control system that applies policies to encoded application network traffic to which it can decode according to the encoding scheme associated with each application.

Although generally described as a gateway 105, the gateway may be any type and form of computing device 100 as described below, such as an appliance, network device, or server. In some embodiments, the gateway 105 establishes or provides a virtual private network connection between a first network 104 and a second network 104'. In one embodiment, the gateway 105 establishes a Secure Socket Layer (SSL) VPN connect between networks 104, 104'. In another embodiment, the gateway 105 establishes a first transport layer connection, such as a TCP connection, between a client 101a-101n and the gateway 105, and establishes a second transport layer connection between the gateway 105 and a server 106a-106n. In another embodiment, the gateway 105 also establishes or provides encrypted sessions between a client 101a-101n and a server 106a-106n. In one embodiment, the gateway 105 may also accelerate the delivery of applications to a client 101a-101n via the transport layer connection(s) using any pooling and/or multiplexing connection techniques at the transport or application layer. In yet another embodiment, the gateway 105 compresses one or more network communications, or portions thereof, between a client 101a-101n and a server 106a-106n. In other embodiments, the gateway 105 may also include a cache for caching any one or more network communications, or portions thereof, between a client 101a-101n and a server 106a-106n.

Although the application charset encoding/inspection system 120 is generally shown deployed in a gateway 105 as in FIG. 1A, the system 120 may also be

deployed in any computing device 100. Referring now to FIG. 1B, for example, the application charset encoding/inspection system 120 may be deployed in any one or more of the clients 101a-101n, such as client 101a. In one embodiment, upon a client's 101 request to access a network 104 via the gateway 105, the gateway may provide the system 120 to install on the client 101. In some embodiments, the system 120 is automatically installed by the client 101 upon receipt from the gateway 105. In another embodiment, the application charset encoding/inspection system 120 may be deployed in any server 106a-106n, such as server 106b. In yet another embodiment, the system 120 may be distributed and have any one or more portions executing on a client 101, gateway 105, and/or server 106. In one embodiment, a plurality of instances of the system 120 may execute on any combination of a client 101, gateway 105, and/or server 106.

FIGs. 1C and 1D depict block diagrams of a computing device 100, and in some embodiments, also referred to as a network device, network appliance or an appliance 100, useful for practicing an embodiment of the application charset encoding/inspection system 120 described herein. As shown in FIGs. 1C and 1D, each computing device 100 includes a central processing unit 102, and a main memory unit 122. As shown in FIG. 1C, a typical computing device 100 may include a visual display device 124, a keyboard 126 and/or a pointing device 127, such as a mouse. Each computing device 100 may also include additional optional elements, such as one or more input/output devices 130a-130b (generally referred to using reference numeral 130), and a cache memory 140 in communication with the central processing unit 102.

The central processing unit 102 is any logic circuitry that responds to and processes instructions fetched from the main memory unit 122. In many

embodiments, the central processing unit is provided by a microprocessor unit, such as: those manufactured by Intel Corporation of Mountain View, California; those manufactured by Motorola Corporation of Schaumburg, Illinois; those manufactured by Transmeta Corporation of Santa Clara, California; those manufactured by International Business Machines of White Plains, New York; or those manufactured by Advanced Micro Devices of Sunnyvale, California. The computing device 100 may be based on any of these processors, or any other processor capable of operating as described herein.

Main memory unit 122 may be one or more memory chips capable of storing data and allowing any storage location to be directly accessed by the microprocessor 102, such as Static random access memory (SRAM), Burst SRAM or SynchBurst SRAM (BSRAM), Dynamic random access memory (DRAM), Fast Page Mode DRAM (FPM DRAM), Enhanced DRAM (EDRAM), Extended Data Output RAM (EDO RAM), Extended Data Output DRAM (EDO DRAM), Burst Extended Data Output DRAM (BEDO DRAM), Enhanced DRAM (EDRAM), synchronous DRAM (SDRAM), JEDEC SRAM, PC100 SDRAM, Double Data Rate SDRAM (DDR SDRAM), Enhanced SDRAM (ESDRAM), SyncLink DRAM (SLDRAM), Direct Rambus DRAM (DRDRAM), or Ferroelectric RAM (FRAM). The main memory 122 may be based on any of the above described memory chips, or any other available memory chips capable of operating as described herein. In the embodiment shown in FIG. 1C, the processor 102 communicates with main memory 122 via a system bus 150 (described in more detail below). FIG. 1C depicts an embodiment of a computing device 100 in which the processor communicates directly with main memory 122 via a memory port 103. For example, in FIG. 1D the main memory 122 may be DRDRAM.

FIG. 1D depicts an embodiment in which the main processor 102 communicates directly with cache memory 140 via a secondary bus, sometimes referred to as a backside bus. In other embodiments, the main processor 102 communicates with cache memory 140 using the system bus 150. Cache memory 140 typically has a faster response time than main memory 122 and is typically provided by SRAM, BSRAM, or EDRAM.

In the embodiment shown in FIG. 1C, the processor 102 communicates with various I/O devices 130 via a local system bus 150. Various busses may be used to connect the central processing unit 102 to any of the I/O devices 130, including a VESA VL bus, an ISA bus, an EISA bus, a MicroChannel Architecture (MCA) bus, a PCI bus, a PCI-X bus, a PCI-Express bus, or a NuBus. For embodiments in which the I/O device is a video display 124, the processor 102 may use an Advanced Graphics Port (AGP) to communicate with the display 124. FIG. 1D depicts an embodiment of a computer 100 in which the main processor 102 communicates directly with I/O device 130b via HyperTransport, Rapid I/O, or InfiniBand. FIG. 1D also depicts an embodiment in which local busses and direct communication are mixed: the processor 102 communicates with I/O device 130a using a local interconnected bus while communicating with I/O device 130b directly.

The computing device 100 may support any suitable installation device 116, such as a floppy disk drive for receiving floppy disks such as 3.5-inch, 5.25-inch disks or ZIP disks, a CD-ROM drive, a CD-R/RW drive, a DVD-ROM drive, tape drives of various formats, USB device, hard-drive or any other device suitable for installing software and programs such as any software 120, or portion thereof, related to an application charset encoding/inspection system 120. The computing device 100 may further comprise a storage device 128, such as one or more hard disk drives or

redundant arrays of independent disks, for storing an operating system and other related software, and for storing application software programs such as any program related to a application charset encoding/inspection system 120. Optionally, any of the installation devices 116 could also be used as the storage device 128.

Furthermore, the computing device 100 may include a network interface 118 to interface to a Local Area Network (LAN), Wide Area Network (WAN) or the Internet through a variety of connections including, but not limited to, standard telephone lines, LAN or WAN links (*e.g.*, 802.11, T1, T3, 56kb, X.25), broadband connections (*e.g.*, ISDN, Frame Relay, ATM), wireless connections, or some combination of any or all of the above. The network interface 118 may comprise a built-in network adapter, network interface card, PCMCIA network card, card bus network adapter, wireless network adapter, USB network adapter, modem or any other device suitable for interfacing the computing device 100 to any type of network capable of communication and performing the operations described herein.

A wide variety of I/O devices 130a-130n may be present in the computing device 100. Input devices include keyboards, mice, trackpads, trackballs, microphones, and drawing tablets. Output devices include video displays, speakers, inkjet printers, laser printers, and dye-sublimation printers. The I/O devices may be controlled by an I/O controller 123 as shown in FIG. 1C. The I/O controller may control one or more I/O devices such as a keyboard 126 and a pointing device 127, *e.g.*, a mouse or optical pen. Furthermore, an I/O device may also provide storage 128 and/or an installation medium 116 for the computing device 100. In still other embodiments, the computing device 100 may provide USB connections to receive handheld USB storage devices such as the USB Flash Drive line of devices manufactured by Twintech Industry, Inc. of Los Alamitos, California.

In further embodiments, an I/O device 130 may be a bridge 170 between the system bus 150 and an external communication bus, such as a USB bus, an Apple Desktop Bus, an RS-232 serial connection, a SCSI bus, a FireWire bus, a FireWire 800 bus, an Ethernet bus, an AppleTalk bus, a Gigabit Ethernet bus, an Asynchronous Transfer Mode bus, a HIPPI bus, a Super HIPPI bus, a SerialPlus bus, a SCI/LAMP bus, a FibreChannel bus, or a Serial Attached small computer system interface bus.

A computing device 100 of the sort depicted in FIGs. 1C and 1D typically operate under the control of operating systems, which control scheduling of tasks and access to system resources. The computing device 100 can be running any operating system such as any of the versions of the Microsoft® Windows operating systems, the different releases of the Unix and Linux operating systems, any version of the Mac OS® for Macintosh computers, any embedded operating system, any network operating system, any real-time operating system, any open source operating system, any proprietary operating system, any operating systems for mobile computing devices or network devices, or any other operating system capable of running on the computing device and performing the operations described herein. Typical operating systems include: WINDOWS 3.x, WINDOWS 95, WINDOWS 98, WINDOWS 2000, WINDOWS NT 3.51, WINDOWS NT 4.0, WINDOWS CE, and WINDOWS XP, all of which are manufactured by Microsoft Corporation of Redmond, Washington; MacOS, manufactured by Apple Computer of Cupertino, California; OS/2, manufactured by International Business Machines of Armonk, New York; and Linux, a freely-available operating system distributed by Caldera Corp. of Salt Lake City, Utah, or any type and/or form of a Unix operating system, among others.

In other embodiments, the computing device 100 may have different processors, operating systems, and input devices consistent with the device. The

computing device 100 can be any workstation, desktop computer, laptop or notebook computer, server, handheld computer, mobile telephone or other portable telecommunication device, media playing device, combination device, purpose-built, special, custom or proprietary device or any other type and/or form of computing or telecommunications device that is capable of communication and that has sufficient processor power and memory capacity to perform the operations of the present invention described herein.

Referring now to FIG. 2, an embodiment of the application charset encoding/inspection system 120 is depicted. The system 120 can reside and/or operate on any type and form of computing device 100, such as a network device, appliance, gateway, client or server device. In brief overview, the system 120 includes a receiver 215, a transmitter 220, a character set engine 225, and a rules or policy engine 250. The receiver 215 and transmitter 220 may be used to receive and send communications via network 104 or between networks 104 and 104'. The character set engine 225 is used to process network communications, such as request, to determine the type and/or form of encoding to be used for an application associated with a network communication. The rule/policy engine 250 applies one or more rules or policies to network communications processed by the system 120. For example, based on the type of encoding determined to be associated with an application by the character set engine 225, the policy engine 250 may control, limit or prevent the network communication from being transmitted by the transmitter 220. In one embodiment, the policy engine 250 enables the system 120 to operate or act as a firewall or a security control device. Additionally, the policy engine 250 may provide policies to determine and control the encoding used and the action to take upon decoded content.

The receiver 215 may comprise software, hardware, or any combination of software and hardware to receive signals via the medium of the device's 100 connection to a network 104. Likewise, the transmitter 220 may comprise software, hardware, or any combination of software and hardware to transmit signals via the medium of the device's 100 connection to a network 104. The network 104 and network connections may include any type of transmission medium between any of the computing devices 100a-100n such as electrical wiring or cabling, fiber optics, electromagnetic radiation or otherwise have any other form of transmission medium capable of supporting the operations described herein. In one embodiment, the receiver 215 receives one or more signals via a first type of medium. In some embodiments, the transmitter 220 transmits one or more signals via a second type of medium. In other embodiments, the receiver 215 and transmitter 220 receive and transmit signals on the same type of medium. In another embodiment, a transceiver includes the receiver 215 and transmitter 220 to receive and transmit signals via a medium.

The device 100 and/or system 120 includes a network stack 210. In some embodiments, the receiver 215 and/or transmitter 220 may include a network stack 210. In other embodiments, the receiver 215 and/or transmitter 220 may include a plurality of networks stacks. In another embodiment, the receiver 215 and/or transmitter 220 interface, integrate or otherwise communicate with one or more network stacks 210. The network stack 210 may comprise any type and form of software, or hardware, or any combinations thereof, for providing connectivity to and communications with a network. In one embodiment, the network stack 210 comprises a software implementation for a network protocol suite. The network stack 210 may comprise one or more network layers, such as any networks layers of the

Open Systems Interconnection (OSI) communications model as those skilled in the art recognize and appreciate. As such, the network stack 210 may comprise any type and form of protocols for any of the following layers of the OSI model: 1) physical link layer, 2) data link layer, 3) network layer, 4) transport layer, 5) session layer, 6) presentation layer, and 7) application layer. In one embodiment, the network stack 210 may comprise a transport control protocol (TCP) over the network layer protocol of the internet protocol (IP), generally referred to as TCP/IP. In some embodiments, the TCP/IP protocol may be carried over the Ethernet protocol, which may comprise any of the family of IEEE wide-area-network (WAN) or local-area-network (LAN) protocols, such as those protocols covered by the IEEE 802.3. In some embodiments, the network stack 210 comprises any type and form of a wireless protocol, such as IEEE 802.11 and/or mobile internet protocol.

In view of an embodiment of a TCP/IP based network 104, in one embodiment, any TCP/IP based protocol may be used, including Messaging Application Programming Interface (MAPI) (email), File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), Common Internet File System (CIFS) protocol (file transfer), Independent Computing Architecture (ICA) protocol, Remote Desktop Protocol (RDP), Wireless Application Protocol (WAP), Mobile IP protocol, and Voice Over IP (VoIP) protocol. In another embodiment, the network stack 210 comprises any type and form of transport control protocol, such as a modified transport control protocol, for example a Transaction TCP (T/TCP), TCP with selection acknowledgements (TCP-SACK), TCP with large windows (TCP-LW), a congestion prediction protocol such as the TCP-Vegas protocol, and a TCP spoofing protocol. In other embodiments, any type and form of user datagram protocol (UDP),

such as UDP over IP, may be used by the network stack 210, such as for voice communications or real-time data communications.

Furthermore, the network stack 210 may include one or more network drivers supporting the one or more layers, such as a TCP driver or a network layer driver. The network drivers may be included as part of the operating system of the computing device 100 or as part of any network interface cards or other network access components of the computing device 100. In some embodiments, any of the network drivers of the network stack 210 may be customized, modified or adapted to provide a custom or modified portion of the network stack 210 in support of any of the techniques of the present invention described herein. In other embodiments, the system 120 is designed and constructed to operate with or work in conjunction with the network stack 210 installed or otherwise provided by the operating system of the device 100.

Still referring to FIG. 2, the character set engine 225 comprises any type and form of logic, functions and operations for determining a charset encoding to associate with an application. The character set engine 225, and any portion thereof, may comprise software, hardware, or any combination of software and hardware. In brief overview, the character set engine 225 comprises a parser 230, an application determination mechanism 235, and an analyzer 240. In some embodiments, the character set engine 225 receives or intercepts a network communication to or from an application, such as an application 110a-110b as illustrated in FIGs. 1A and 1B. For example, an application 110a on a client 100a may communicate a request to a server 110d via network 104. In one embodiment, the character set engine 225 is interfaced or otherwise in communication with the receiver 215, transmitter 220, and/or network stack 220.

In some embodiments, the parser 230 of the character set engine comprises logic, functions or operations to parse any network communication received or intercepted by the device 100. In one embodiment, the parser 230 identifies, parses, extracts, and/or interprets any portion of a network communication. In one embodiment, the parser 230 parses any application layer protocol communication, such as the HyperText Transfer Protocol (HTTP), the Extensible Markup Language (XML) protocol or Simple Mail Transfer protocol (SMTP). For example, the parser 230 may identify and parse, or extract, one or more fields submitted via a form, such as an HTTP form submission, from a client 100a to a server 100d. In another example, the parser 230 may identify, and parse or extract any attributes, cookies, name-value pairs, URLs, data strings, objects, or any part of a request, such as an HTTP form submission.

In one embodiment, the parser 230 identifies in the request an attribute, header, field or data element identifying a type of content, such as text, image, mixed-data types, etc. For example, in an embodiment of an HTTP protocol, a content-type header is used to specify the media type and subtype of data in the body of a message and to specify the native representation of such data. In some embodiment, the parser 230 identifies a portion of the request is encoded. For example, in an embodiment of an HTTP protocol, a content-type header may identify a portion of a URL request is encoded. In another embodiment, the parser 230 identifies in the request an attribute identifying a charset to use for encoding or decoding a portion of the content of the request.

In one embodiment, the parser 230 parses any portion of a transport layer protocol packet, such as a TCP or UDP packet. In some embodiments, the parser 230 identifies and parsers any of the following from a transport layer network packet: 1)

source internet protocol address, 2) destination internet protocol address, 3) source port, 4) destination port, 5) any data of the header and/or payload of the packet identifying the protocol or protocols; and 5) any fields of the packet header. In another embodiment, the parser 230 creates or provides an object model representation, or object based application programming interface, for any of the parsed network communications or identified elements of the network communications.

The application determination mechanism 235 comprises any logic, function and/or operations to determine an application associated with a network communication, such as a message or a request. In one embodiment, the application determination mechanism 235 is in communication with or interfaced to the parser 230 and obtains one or more parsed information from the network communication. In some embodiments, the application determination mechanism 235 determines from the request or any information from or representing the request, the type, name or identification of an application associated with the request. In one embodiment, the application determination mechanism 235 identifies the type of encoding for the application and/or the request. The encoding information may be used for decoding, inspecting, analyzing or otherwise processing the request.

In some embodiments, the application determination mechanism 235 is configured to associate a name, type or identifier of an application with one or more data elements of a network communication, such as a source internet protocol address and/or port, or destination internet protocol address and/or destination port. For example, a network communication from a particular client may be associated with an application. In another example, the system 120 may associate a network communication to one or more servers in an internet protocol range or using a port or

port range. In other embodiments, the application determination mechanism 235 is configured to associate a name, type or identifier of an encoding scheme, character encoding set, or encoding mechanism with one or more data elements of a network communication, such as any parsed fields provided by the parser 230. In yet another embodiment, the application determination mechanism 235 determines the type, name or identification of the application and/or encoding scheme via parsing of the network communication, such as by information carried by the payload of a network packet.

In some embodiments, the application determination mechanism 235 uses a database, file, object, data structure or other information storage medium to store configuration information associating a network communication, or any portion thereof, to an application and/or encoding scheme. For example, an application may be mapped to one or more internet protocol addresses and/or ports. In another example, the application determination mechanism 235 may lookup the application associated with or based on one or more data elements identified in the network communication from any type and form of lookup table. In one embodiment, the application determination mechanism 235 is configurable by one or more users via any type and form of interface, such as a command line interface or graphical user interface. In another embodiment, the application determination mechanism 235 is configured via an application programming interface by another program, script, application or system.

Upon determining the application associated with a network communication, such as a request, the system 120 determines, identifies or otherwise obtains the encoding type to be used to decode any encoded portion of the network communication. In one embodiment, the system 120, such as the application determination mechanism 235 and/or analyzer 240, identifies the encoding type from

any portion or data element of the network communication itself. For example, the system 120 may identify the encoding type from any parsed elements of the network communication, such as data in the payload of a network packet. In another embodiment, the system 120 identifies or obtains the encoding type for the application from a query or lookup, such as via an application programming interface, into a table, database, file, object, data structure or other storage medium or configuration mechanism having such information.

In yet another embodiment, the system 120 identifies or obtains the encoding scheme for the application from the rules/policy engine 250. For the example, the application determination mechanism 235 and/or analyzer 240 may query the policy engine 250 to obtain the encoding type for a given application. In some embodiments, an application may have a plurality of encoding types associated with it based on temporal information, client information, user information, device information, status of the network, status of any system, historical information, and/or statistical information. In one embodiment, the system 120 requests from the rules/policy engine 250 the encoding type for an application based on one or more of the above types of information. For example, a first application may use or may be allowed to use a first encoding type on a first day or time of the week and a second encoding type on a second day or time of the week. In one example, the system 120 may query the policy engine 250 with the identified application and temporal information to determine the encoding type to be used for processing the network communication of the application.

The analyzer 240 comprises any logic, function and/or operations to analyze the network communication, or any portion thereof. In one embodiment, the analyzer 240 decodes the portion of the network communication, such as a request, that is

encoded in a charset. In some embodiments, the analyzer 240 may obtain the encoding scheme to use for an application from any other portion of the character set engine 225, such as the parser 230 or the application determination mechanism 235. In other embodiments, the analyzer 240 obtains the encoding scheme for an application from the rules/policy engine 250. In one embodiment, the parser 230 or application determination mechanism 235 provides the network communication to the analyzer 240 with the encoded portion of the network communication decoded using the encoding type for the application.

In one embodiment, the analyzer 240 uses the identification of the application and/or the associated encoding type to inspect or analyze content of the network communication. In some embodiments, the analyzer 240 performs uni-directional and bi-directional analysis on a stream of network traffic received by the system 120. For example, the analyzer 240 may perform a deep stream inspection on each of the network packets. In other embodiments, the analyzer 240 inspects and analyzes the HTTP and HTML header and payload. In one embodiment, the system 120 can perform full HTML parsing, such as via parser 230, and the analyzer 240 can inspect and analyze any portion of an HTML communication. In yet another embodiment, the analyzer 240 identifies, maintains and tracks sessions, and states of sessions, of network traffic received and processed by the system 120.

Still referring to FIG. 2, the system 120 may also include a rule/policy engine 250 for applying a set of one or more policies based on the inspection, filtering or analysis of a network communication. In one embodiment, the policy engine 250 comprises a policy regarding the date, time or schedule by which an application can access the network 204. In another embodiment, the policy engine 250 comprises a policy regarding the date, time or schedule by which an application can be used by an

identified computing device 100 or an identified user. In still another embodiment, the policy engine 250 comprises a policy regarding the date, time or schedule by which an encoding scheme is to be used or can be used for an application. For example, a user may configure a rule or policy of the system 120 to allow a first application to use a first encoding scheme during a first day of the week or during a first specified time range and to use a second encoding scheme during a second day of the week or during a second specified time range.

In one embodiment, the system 120 comprises an end-point detection and scanning mechanism, which identifies and determines one or more attributes or characteristics of the client. For example, the system 120 may identify and determine any one or more of the following client-side attributes: 1) the operating system and/or a version of an operating system, 2) a service pack of the operating system, 3) a running service, 4) a running process, and 5) a file. The system 120 may also identify and determine the presence or versions of any one or more of the following on the client: 1) antivirus software, 2) personal firewall software, 3) anti-spam software, and 4) internet security software. The policy engine 250 may have one or more policies based on any one or more of the attributes or characteristics of the client or client-side attributes. In some embodiments, the policy engine 250 may specify the type of encoding scheme associated with an application or the type of decoding to use for an application based on any client attributes. For example, the policy engine 250 may comprise a policy if the client is running a specific language version of the operating system that an encoding scheme associated with the specific language is used for decoding encoded requests from an application running on that client.

In some embodiments, the rules/policy engine 240 comprises one or more application firewall or security control policies for providing protections against

various classes and types of web or Internet based vulnerabilities, such as one or more of the following: 1) buffer overflow, 2) CGI-BIN parameter manipulation, 3) form/hidden field manipulation, 4) forceful browsing, 5) cookie or session poisoning, 6) broken access control list (ACLs) or weak passwords, 7) cross-site scripting (XSS), 8) command injection, 9) SQL injection, 10) error triggering sensitive information leak, 11) insecure use of cryptography, 12) server misconfiguration, 13) back doors and debug options, 14) website defacement, 15) platform or operating systems vulnerabilities, and 16) zero-day exploits. In an embodiment, the system 120 provides HTML form field protection in the form of inspecting or analyzing the network communication for one or more of the following: 1) required fields are returned, 2) no added field allowed, 3) read-only and hidden field enforcement, 4) drop-down list and radio button field conformance, and 5) form-field max-length enforcement. In some embodiments, the system 120 ensures cookies are not modified. In other embodiments, the system 120 protects against forceful browsing by enforcing legal URLs.

In still yet other embodiments, the system 120 protects any confidential information contained in the network communication. The system 120 may inspect or analyze any network communication in accordance with the rules or policies of the engine 250 to identify any confidential information in any field of the network packet. In some embodiments, the system 120 identifies in the network communication one or more occurrences of a credit card number, password, social security number, name, patient code, contact information, and age. The encoded portion of the network communication may comprise these occurrences or the confidential information. Based on these occurrences, in one embodiment, the system 120 may take a policy action on the network communication, such as prevent transmission of the network

communication. In another embodiment, the system 120 may rewrite, remove or otherwise mask such identified occurrence or confidential information.

With the per application encoding identification and decoding functionality of the system 120, the analyzer 240 and the policy engine 250 may apply application firewall and security control to encoded network communications of a plurality of applications, each using one or more different encoding types concurrently or subsequently. As such, the rules and policy configured in the rules/policy engine 250 can be applied at both the granularity of a type, a name or instance of an application and the encoding scheme associated with such application as determined in accordance with the operations of the system described herein. Furthermore, the system 120 allows for the analysis of encoded portions of network communications that could not be decoded, inspected, and analyzed without knowing the encoding scheme. By doing so, the system 120 can apply policies, such as application firewall and security policies, to the encoded portions of network communications, such as a request having url encoded content, on a per application and/or per encoding scheme basis.

Although the parser 230, application determination mechanism 235 and the analyzer 240 are illustrated as included in the character set engine 225 in FIG. 2, any of the parser 230, application determination mechanism 235 or the analyzer 240, or any portion thereof, may reside, operate or execute in any portion of the device 100 or application charset encoding/decoding system 120. Additionally, although shown as a single logical entity or component, the application charset encoding/inspection system 120 may also operate in a distributed manner, with a first portion running on a first device 100a, such as a client, and a second portion running on a second device 100b, such as a server or gateway. In yet another embodiment, a plurality of application

charset encoding/inspection systems, e.g., 120, 120', etc., may operate in cooperation or in conjunction with each other to provide the functionality and techniques described herein for one or more applications, gateways, clients, or servers.

The operations of the application charset encoding/inspection system 120 may support any type and form of encoding scheme or character encoding set (charset). In some embodiments, the application charset encoding/inspection system 120 operates with any type and form of Unicode scheme, including, by way of example, UTF-7, UTF-8, CESU-8, UTF-16/UCS-2, UTF-32/UCS-4, UTF-EBCDIC, SCSU, Punycode, GB 18030. Unicode is a character encoding scheme or set allowing characters from Western European, Eastern European, Cyrillic, Greek, Arabic, Hebrew, Chinese, Japanese, Korean, Thai, Urdu, Hindi and all other major world languages, living and dead, to be encoded in a single character set. In one embodiment, the character set is 16-bits. In other embodiments of an encoding scheme, the character set may be 6, 7, 8, 10, 12, 20, 24, 32, or 64-bits or any other number of bits. The Unicode specification also includes standard compression schemes and a wide range of typesetting information required for worldwide locale support. In one embodiment, the application charset encoding/inspection system 120 operates using ISO 10646 family of standards, which defines several character encoding forms for the Universal Character Set. In some embodiments, the application charset encoding/inspection system 120 uses an ASCII encoding scheme. In other embodiments, the application charset encoding/inspection system 120 operates on non-ASCII encoded requests, or portions thereof. In another embodiment, the application charset encoding/inspection system 120 performs its operations on a request using the ANSI or WGL4 character sets.

In some embodiments, the application charset encoding/inspection system 120 operates with an encoding scheme, character encoding set or charset that supports or is used to represent any type and form of language, such as a language of Japan, Korean, Russia, or China, including any dialects and differences therein, and any one or multiple encoding schemes used therefore. By way of example, and not intending to be in any way limiting or exclusive, for Russian, the system 120 may operate with the following types of encoding schemes or charsets: 1) Cyrillic (CP1251), 2) KOI8r, KOI-8 Alternative, KOI-8 Unified, or KOI-8RU, 3) Unicode or UTF-8, 4) DosCyrillicRussian (CP866), 5) ISO-8859-5, and 6) ECMA-Cyrillic (ISO-IR-111).

By way of example, and not intending to be in any way limiting or exclusive, for the Japanese language, the system 120 may operate with the following encoding schemes, character encoding sets, or charsets: 1) utf-8, 2) JIS (Japanese Industry Standard), 3) shift_jis (also known as SJIS, X-SJIS or MS Kanji), 4) EUC/EUC-JP (extended Unix code), 5) EBDIC, 6) ISO2022/ISO2022-JP, 7) ANSI Z39.64, 8) CCCII, 9) DEC Kanji, 10) GTcode, 11) IBM DBCS, 12) JEF (Japanese Extended Features), 13) CCCII, 14) ISO-8850, 15) JIS X 0201 (JISROMAN), 16) JIS X 0208 (JIS C 6226), 17) JIS X 0212, JIS X 0213, or JIS X 0221, and 18) Mojikyo. By way of example, and not intending to be in any way limiting or exclusive, for Korean, the system 120 may operate with any of the following encoding schemes or charsets: 1) utf-8, 2) EUC, or EUC-KR, 3) KEIS, 4) ANSI Z39.64, 5) ISO-2022, or ISO-2022-KR, 6) CCCII, 7) Unified Hangul Code (CP949), 8) GB 12052, 7) IBM DBCS, 8) JOHAB, 9) KS C 5601, 10) KS C 5636 (KS ROMAN), 11) KS C 5657, 12) KS C 5700, and 13) Mojikyo.

By way of example, and not intending to be in any way limiting or exclusive, for the Chinese language, the system 120 may operate with the following encoding

schemes or charsets: 1) utf-8, 2) ANSI Z39.64, 3) Big5, Big5+, Big5ETen, or Big5-HKSCS, 4) CCCII, 5) CNS 11643, 6) GBK (CP936), 7) CP90, 8) EUC/EUC-CN/EUC-TW, 9) GB 12050/12052, 10) GB13000-1, 11) GB13134, 12) GB16959, 13) GB18030, 14) GB1988, 15) GB2312, 16) GB7589, 17) GB7590, 18) GB8045, 19) GB/T 12345, GB/T 13131, or GBT/13132, 20) HZ, 21) ISO2022/2002-CN/CN-EXT, and 22) Mojikyo.

The system 120 may operate with a plurality of languages and a plurality of encoding schemes in use at any one time, subsequently to or concurrently with each other. In some embodiments, the system 120 may operate using the same encoding scheme for a plurality of languages, such as using the same encoding for Japanese, Korean and Chinese, or with different encoding schemes for each of a plurality of languages, such as different encoding schemes for each of Japanese, Korean and Chinese.

Referring now to FIG. 3, an embodiment of a method for determining an encoding type associated with an application or otherwise applying the techniques of the system 120 is depicted. In brief overview of method 300, at step 310, the application charset encoding/inspection system 120 receives a request. At step 315, the system 120 determines for which application of a plurality of applications to which the request corresponds. At step 320, the system 120 identifies the encoding scheme associated with the determined application. At step 325, the system 120 uses the identified encoding scheme to decode, inspect and/or analyze the request. Upon decoding and analyzing the request, at step 330, the system 120 may apply one or more policies to the request.

In further details, at step 310, the system 120 may receive or intercept a network communication, such as request, by any means and/or mechanism. In one

embodiment, the receiver 210 receives the request from the client 101. In another embodiment, the receiver 210 intercepts the request from the network stack 210 as it is communicated to or from a server 106. In some embodiments, the system 120, such as the receiver 210, comprises a network driver, filter or hooking mechanism for intercepting the request in the network stack 210. In some embodiments, a gateway 105 deploying the system 120 receives or intercepts the request. In yet another embodiment, the client 101 is configured to send requests to the gateway 105, acting as a proxy for the client 101. In other embodiments, a client 101 or server 106 deploying the system 120 receives or intercepts a request from the client 101. In some embodiments, the system 120 receives a cached form page. In yet another embodiment, the system 120 uses a cached form page stored in a cache of the system 120, or a device embodying the system 120, such as the gateway 105.

In some embodiments, the system 120 does not have prior knowledge of the encoding scheme used by the request. In one embodiment, the request itself does not identify the encoding scheme used for the encoded portion of the request. For example, in one embodiment, the request includes a submission of a form, such as an HTML form, using the form-url-encoded content type. In another embodiment, the request does not provide a tag to identify the character encoding. In yet other embodiments, the request includes an identification of the encoding system. In still another embodiment, the system 120 understands the encoding scheme for the request by guessing the encoding type using heuristic rules or logic. In some embodiments, the system 120 may determine the encoding scheme based on the behavior of the application or the client. In one embodiment, the system determines the encoding scheme based on using an encoding scheme known between the system 120, gateway 105, server 106, and/or client 101.

At step 315, the system 120 determines one of a plurality of applications to which the request corresponds. In one embodiment, the application determination mechanism 235 determines the application from one or more data elements identified and/or parsed from the request, such as by the parser 230. In some embodiments, the application determination mechanism 235 determines the application generating or associated with the request by mapping an internet protocol address and/or port to a lookup of the corresponding application from a database, table, file, object, data structure or other storage medium. In one embodiment, the application determination mechanism 235 determines the application from a data element in a payload of the request identifying the application by name, type or instance.

At step 320, the system 120, such as the charset engine 225 identifies the encoding scheme or charset for the application determined at step 315. In one embodiment, the charset engine 225, such as via the application determination mechanism 235 or the analyzer 240, queries or performs a lookup of the encoding scheme for the application from a database, file, table, object, data structure or other storage medium mapping the application to one or more encoding schemes. In yet another embodiment, the charset engine 225 determines the encoding scheme for the application from any portion of the request. In one embodiment, the charset engine 225 identifies the encoding scheme for one or more data elements identified or parsed by the parser 230. In some embodiments, the charset engine 225 identifies the encoding scheme for the application from a cache, memory or storage element storing the encoding scheme associated with the application. For example, in one embodiment, the charset engine 225 tracks the previously used encoding scheme for the application. In yet other embodiments, the charset engine 225 identifies the encoding scheme for the application from a network communication, such as a

response to a client request, having information identifying the charset. For example, in one embodiment, the system 120 identifies and parses such information from the server's network communication.

In some embodiments, the charset engine 225, such as via the application determination mechanism 235 or the analyzer 240, obtains the encoding scheme to use for the application from the rules/policy engine 250. For example, the policy engine 250 may specify the encoding scheme to use for an application based on any temporal information related to the request, such as date and time. In another example, the policy engine 250 may specify the encoding scheme to use for an application based on any system information or attributes of the client communicating the request. In one embodiment, the system 120 performs an end-point detection and scan of the client and determines one or more attributes or characteristics of the client. The policy engine 250 may apply one or more policies to the request, or to the decoding of the encoded portion of the request, based on any attributes of the client. For example, the policy engine 250 may specify to use a first type of encoding scheme for an application if the client is running a certain type of operating system.

At step 325, the system 120 uses the identifying encoding scheme to decode the encoded portion of the request. In one embodiment, the charset engine 225, such as via the parser 230, application determination mechanism 235, and/or analyzer 240, applies the identified encoding scheme to the encoded portion of the request. As such, the analyzer 240 decodes the encoded portion into a data element, text or string that can be inspected or analyzed by the analyzer 240. For example, in one embodiment, the decoded portion of the request may form an SQL or other type of command to be executed on a server 106. In some embodiments, the analyzer 240 inspects or analyzes the request including the decoded content to determine if the

request meets or violates any of the rules and/or policies configured via the rules/policy engine 250. As discussed above in conjunction with FIG. 2, the analyzer 240 may perform any logic, function and operations such as bi-directional analysis, deep stream inspection, HTML inspection, session state management, HTML form field protection, cookie poisoning protection, forceful browser protection, and web vulnerabilities protection.

At step 330, the system 120 applies one or more rules or policies to the request based on analysis of the decoded request. In one embodiment, if the request does not satisfy a policy of the policy engine 250 for transmission on the network 104 (or further transmission on the network 104), the system 120, such as a system 120 deployed in gateway 105, may reject or drop the request. In another embodiment, the system 120 may quarantine the application or the user of the application if the request fails a policy. In yet another embodiment, the system 120 may downgrade or limit network access of the application if the request fails a policy. In other embodiments, the system 120 may disconnect the client's connection to the network 104, such as disconnecting a client's SSL VPN connection. In some embodiments, the system 120 may disconnect or terminate the application session if the request fails or does not satisfy a policy.

Although the techniques of method 300 are generally described above in the context of a request from an application, the method 300 can be performed for a plurality of applications subsequently and/or concurrently to each other. The system 120 may receive or intercept a plurality of requests at step 310 from different applications, each having an encoded portion using the same or different encoding scheme as another application. For example, the system 120 may be deployed on a gateway 105 servicing a plurality of clients and applications. For each request of the

plurality of requests, the system 120 determines the application associated with the request at step 315, the charset encoding to be used for the application at step 320, decodes the encoded portion of the request and analyzes the decoded request at step 325, and applies any associated policies to the request at step 330. As such, in some embodiments, the system 120 performs the techniques of method 300 on a per application basis, and applies the associated encoding scheme on a per request basis. In one embodiment, the system 120 may use a first encoding scheme for an application on a first request, and may use a second and different encoding scheme for the same application on a second and subsequent request.

EXAMPLE

The application encoding/inspection system 120 may be deployed in gateway or network appliance for an enterprise network 104 having a plurality of applications using different charsets or encoding schemes. For example, the gateway 105 may be deployed as an application firewall and security control device for a corporate network having Japanese language users. A first application 110a on a first client 101a may use a first charset of UTF-8. A second application 110b on the first client 101a, or on a second client 101b, may use a second charset of JIS. A third application 110c on either the first client 101a or second client 101b, or yet on a third client 101c, may use a third charset of MS Kanji.

Each of the first application 110a, second application 110b, and third application 110c submits one or more requests via the gateway 105 to one or more servers 106a-106n. The applications 110a-110c may comprise web browsers submitting HTTP, HTML and/or XML requests to a web server 106a-106n. Any one or more of these request may comprise a form-url-encoded submission in which the

identification of the charset is not part of the submitted data. Some of the requests may be generated, entirely or otherwise, from a javascript or other script on the browser. Also, one or more of the request may be generated using AJAX, or Asynchronous JavaScript and XML, based technology. As such, for any one or more these requests, the gateway 105 may not have prior knowledge of the charset used for encoding a portion of the request.

In view of the structure, functions and operations of the system 120 described above, the gateway 105 can apply application firewall and security control policies to the encoded portion of the request. For each received request, the gateway 105 determines the application associated with the request and the encoding scheme associated with the application. In this example, the gateway 105 determines a first request is received from the first application 110a, and identifies the UTF-8 encoding scheme as associated with the first application 110a. The gateway 105 decodes the first request with the encoding scheme of UTF-8, analyzes the decoded request, and applies any policies to the request. The gateway 105 determines a second request is received from the second application 110b, and identifies the JIS encoding scheme as associated with the second application 110b. The gateway 105 decodes the first request with the identified encoding scheme of JIS, analyzes the decoded request, and applies any policies to the request. Likewise, the gateway 105 determines a third request is received from the third application 110c, and identifies the MS Kanji encoding scheme as associated with the third application 110c. The gateway 105 decodes the first request with the identified encoding scheme of MS Kanji, analyzes the decoded request, and applies any policies to the request.

In some cases, an application 110a may switch to or otherwise use a different encoding scheme, such as upon starting of another instance during another time

period or another day. For example, the first application 110a may use the JIS charset instead of UTF-8 during this second instance. In these cases, the gateway 105 determines that a subsequent request is received from the first application 110a, and identifies the UTF-8 encoding scheme as associated with the first application 110a. For example, the policy engine 250 may identify the UTF-8 encoding scheme should be used for the first application 110a during a specified time period. The gateway 105 then decodes this subsequent request with the identified encoding scheme of UTF-8, analyzes the decoded request, and applies any policies to the request.

With the gateway 105 described herein, the gateway can decode, analyze and apply policies to requests from a plurality of applications using different encoding schemes. The gateway 105 offers great flexibility in providing a per application and per request decoding mechanism to apply policies to an environment deploying a plurality of different charset encoded applications, such as one may find in a network environment deployed to users of an application in a language of Japanese, Korean, Russian or Chinese. The gateway 105 enables the applying of application firewall and security control devices to differently encoded network communications to protect the network environment from vulnerabilities and security concerns that may be found in encoded content.

Many alterations and modifications may be made by those having ordinary skill in the art without departing from the spirit and scope of the invention. Therefore, it must be expressly understood that the illustrated embodiments have been shown only for the purposes of example and should not be taken as limiting the invention, which is defined by the following claims. These claims are to be read as including what they set forth literally and also those equivalent elements which are

insubstantially different, even though not identical in other respects to what is shown and described in the above illustrations.

We claim:

1. A method for determining the character encoding used to use to decode a request, the method comprising the steps of:
 - (a) receiving a request;
 - (b) determining to which one of a plurality of application programs the request corresponds;
 - (c) identifying a character encoding associated with the determined application program; and
 - (d) using the identified character encoding to inspect the request.
2. The method of claim 1 wherein step (b) comprises determining to which one of the plurality of application programs the request corresponds from an attribute of the request.
3. The method of claim 2 wherein the attribute comprises one of the following: a source identifier; a destination identifier; a port identifier; a protocol identifier; header information; or a Uniform Resource Locator address.
4. The method of claim 1 wherein step (b) comprises determining to which one of the plurality of application programs the request corresponds using a cookie included in the received request.
5. The method of claim 1 wherein step (c) comprises identifying the character encoding associated with the determined application program using a file containing associations between character encodings and applications.
6. The method of claim 1 wherein step (c) comprises identifying the character encoding associated with the determined application program using a database containing associations between character encodings and applications.
7. The method of claim 1 further comprising the steps of:
 - (a) receiving a second request;

(b) determining a second one of the plurality of application programs to which the second request corresponds; and

(c) identifying a second character encoding associated with the determined second application program.

8. The method of claim 1 wherein step (a) comprises receiving a request generated by a client.

9. The method of claim 8 wherein step (b) comprises determining to which one of the plurality of application programs the request corresponds using an attribute of the client.

10. The method of claim 1 wherein step (a) comprises receiving a request based on a cached form page.

11. A gateway capable of determining the character encoding to be used to decode a request, the system comprising:

a receiver in communication with a client via a network and receiving a request from the client;

a character set engine in communication with the receiver, the character set engine identifying a character encoding associated with a received request responsive to the application to which the request is directed and using the identified character set to inspect the request.

12. The gateway of claim 11 wherein the receiver communicates with a plurality of clients.

13. The gateway of claim 11 wherein uses one of the following contained in the request to determine an application program to which the request is directed: a source identifier; a destination identifier; a port identifier; a protocol identifier; header information; or a Uniform Resource Locator address.

14. The gateway of claim 11 wherein the character set engine comprises a database associating character encodings and applications.

15. The gateway of claim 11 wherein the character set engine comprises a file associating character encodings and applications.

16. A method for inspecting by a gateway received from a client a request having an encoded portion, the method comprising the steps of:

- (a) receiving, by a gateway, a request from an application program on a client;
- (b) determining, by the gateway, to which one of a plurality of application programs the request corresponds;
- (c) identifying, by the gateway, a character encoding associated with the determined application program; and
- (d) decoding, by the gateway, a portion of the request using the identified character encoding; and
- (e) inspecting, by the gateway, the decoded portion of the request.

17. The method of claim 16 comprising determining, by the gateway, to which one of the plurality of application programs the request corresponds using an attribute of the request.

18. The method of claim 16 comprising determining, by the gateway, to which one of the plurality of application programs the request corresponds using an attribute of the client.

19. The method of claim 16 comprising applying, by the gateway, a policy to the request based on inspection of the decoded portion of the request.

20. The method of claim 16, comprising

- (f) receiving, by the gateway, a second request from a second application program on one of the client or a second client;
- (g) determining, by the gateway, to which one of the plurality of application programs the second request corresponds;
- (h) identifying, by the gateway, a second character encoding associated with the determined application program; and
- (i) decoding, by the gateway, a portion of the second request using the identified second character encoding.

21. The method of claim 20 comprising inspecting, by the gateway, the decoded portion of the second request.

22. The method of claim 21 comprising applying, by the gateway, a policy to the second request based on inspection of the decoded portion of the second request.

1/6

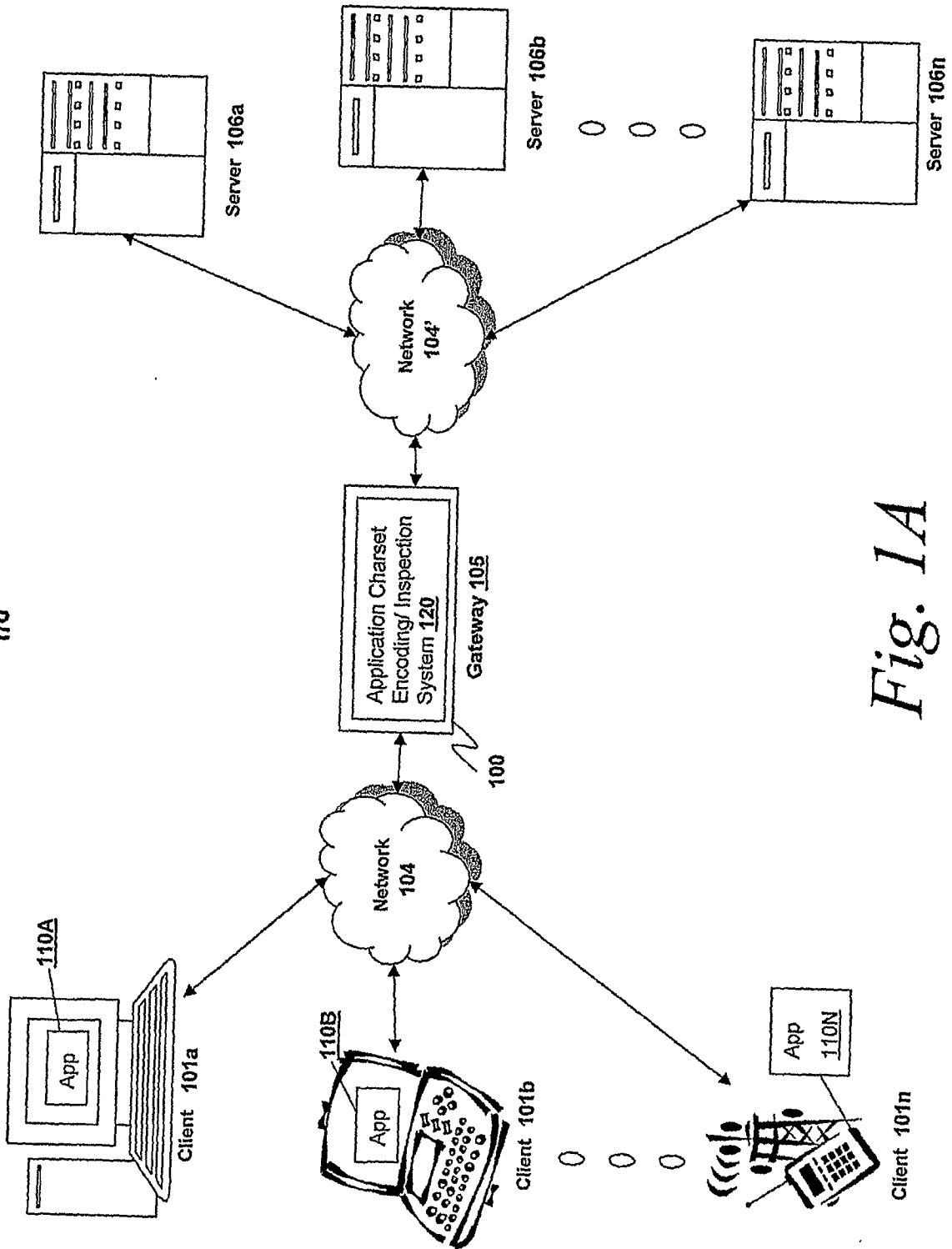


Fig. 1A

2/6

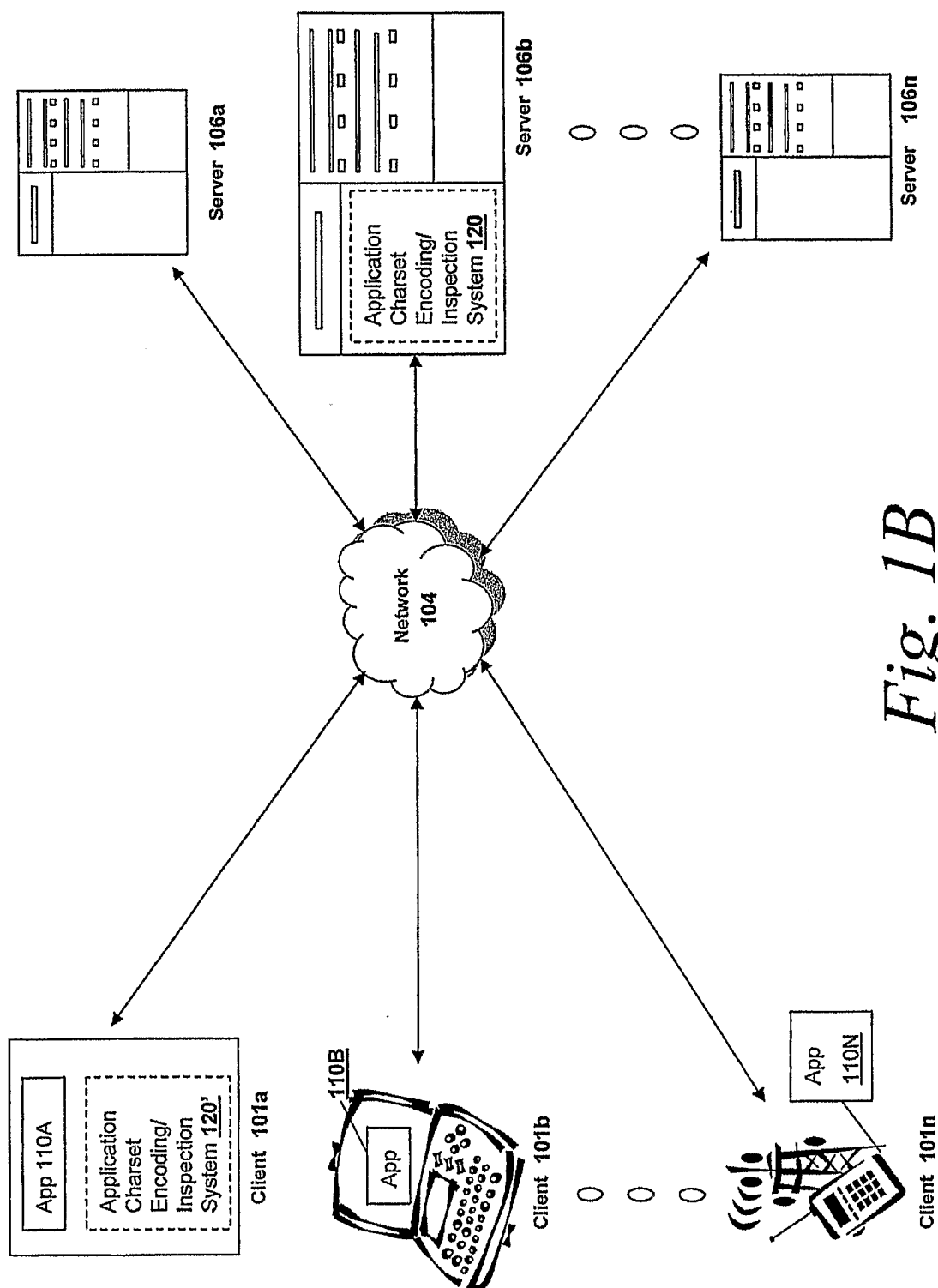
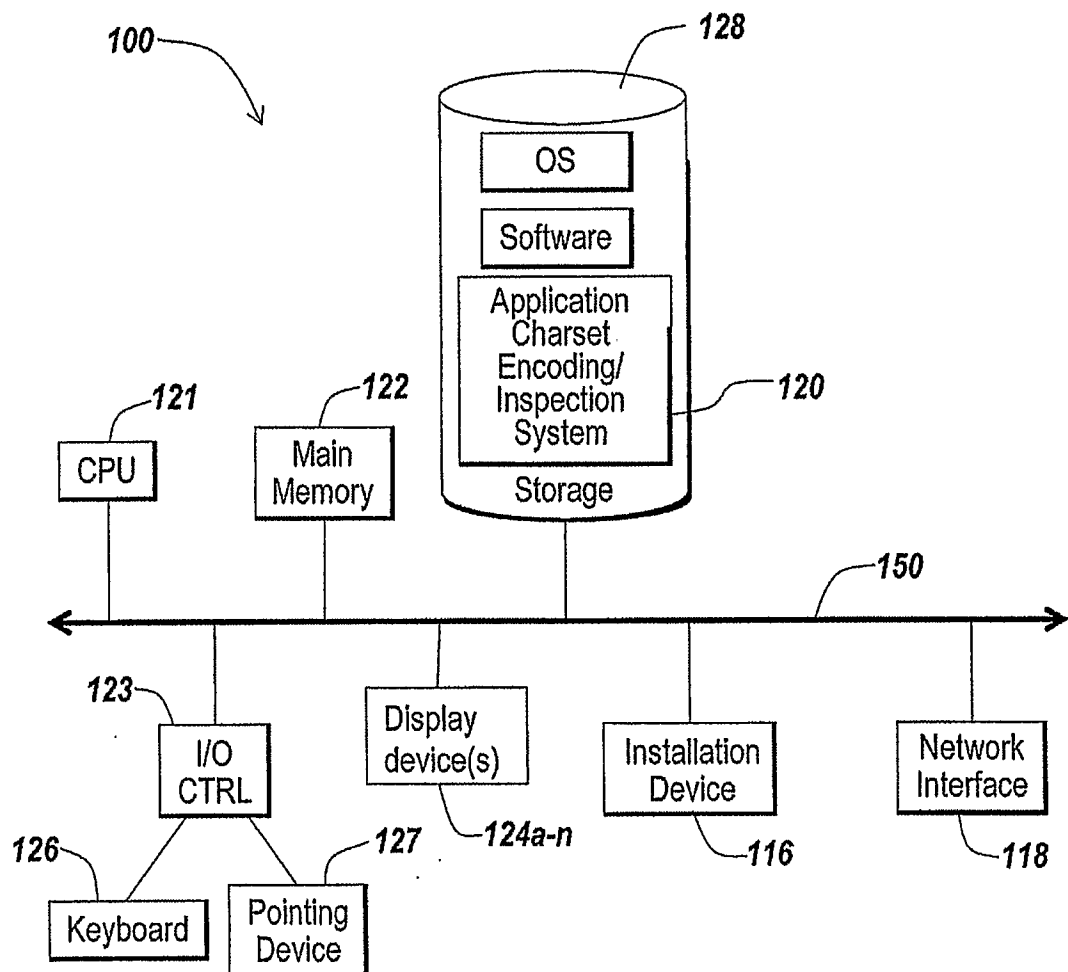
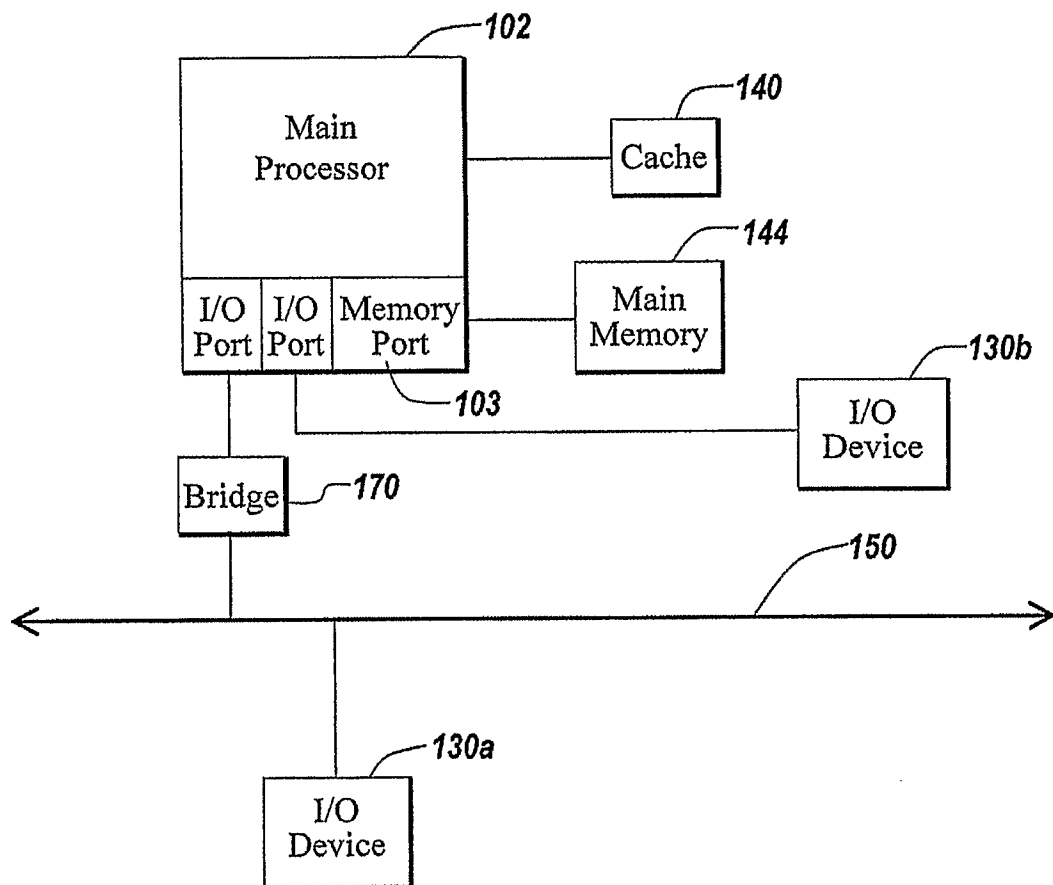


Fig. 1B

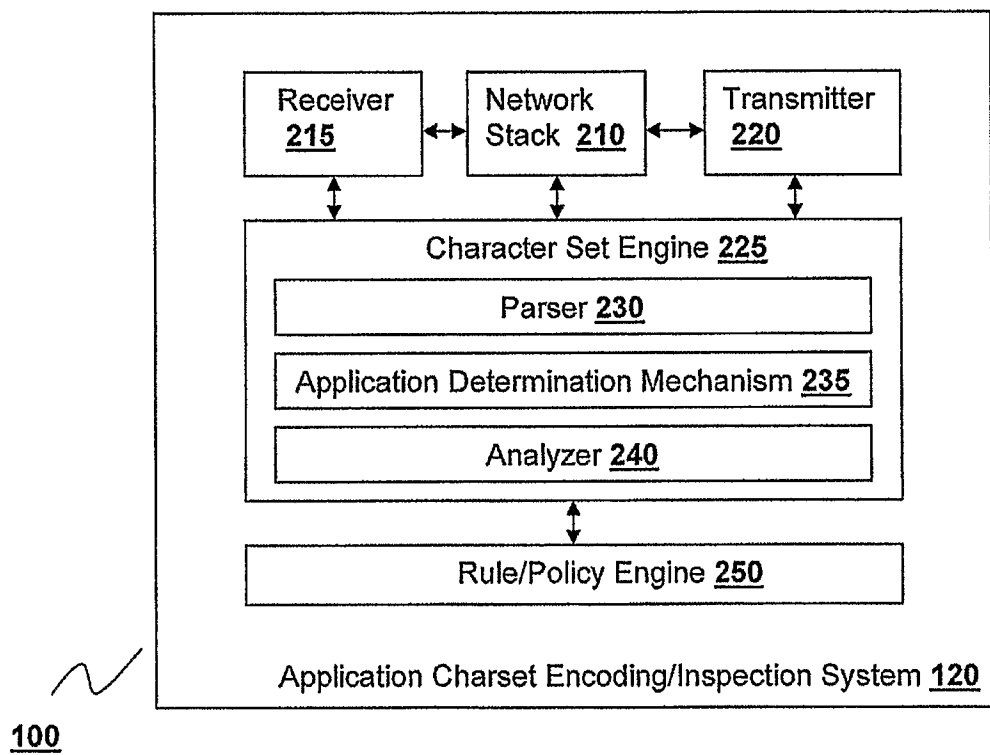
3/6

*Fig. 1C*

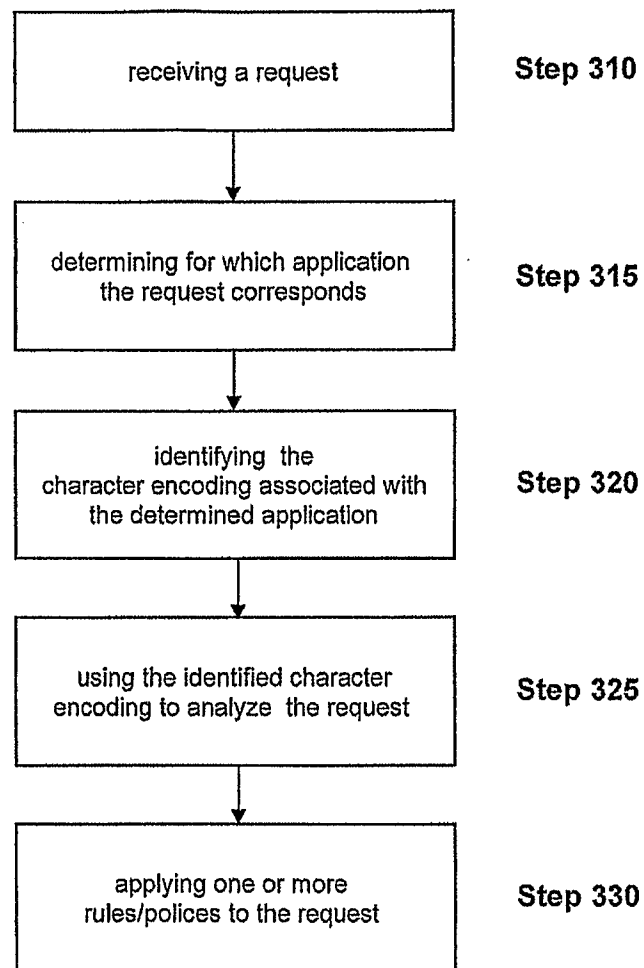
4/6

*Fig. 1D*

5/6

*Fig. 2*

6/6

300*Fig. 3*

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/021067

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2002/191795 A1 (WILLS FERGUS M [IE]) 19 December 2002 (2002-12-19) paragraph [0008] paragraph [0027] paragraph [0031] paragraph [0035] paragraph [0075] figures 2b,6	1-22
A	US 2004/073811 A1 (SANIN ALEKSEY [US]) 15 April 2004 (2004-04-15) paragraph [0016] paragraph [0040] paragraph [0056]	1-22
A	WO 01/33752 A (MEASURECAST INC [US]) 10 May 2001 (2001-05-10) page 14, line 17 - page 16, line 18 ----- -/-	1-22

☒ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier document but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

- "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- "&" document member of the same patent family

Date of the actual completion of the international search

13 February 2007

Date of mailing of the international search report

20/02/2007

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Authorized officer

Cankaya, Sukru

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/021067

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	GB 2 352 850 A (IBM [US]) 7 February 2001 (2001-02-07) page 4, line 4 - page 5, line 20 -----	1-22

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2006/021067

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2002191795 A1	19-12-2002	EP 1313286 A2 US 2005232191 A1	21-05-2003 20-10-2005
US 2004073811 A1	15-04-2004	AU 2003279109 A1 WO 2004036426 A1	04-05-2004 29-04-2004
WO 0133752 A	10-05-2001	AU 1357701 A	14-05-2001
GB 2352850 A	07-02-2001	JP 3407277 B2 JP 2000311135 A US 2003158889 A1	19-05-2003 07-11-2000 21-08-2003